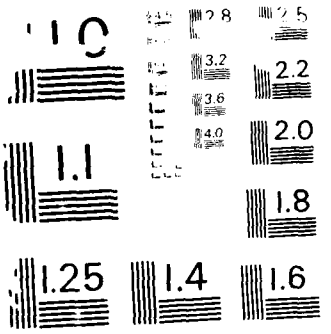
AD-A192 925 SAGUARO: A DISTRIBUTED OPERATING SYSTEM BASED ON POOLS OF SERVERS(U) ARIZONA UNIV TUCSON G ANDREWS ET AL. 1/1
25 MAR 88 AFOSR-TR-88-0400 AFOSR-87-0072

UNCLASSIFIED F/G 25/5 NL



TEST CHART

DTIC FILE COPY

(2)

AD-A192 925

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY: ---	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 88-0408
6a. NAME OF PERFORMING ORGANIZATION: University of Arizona; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR
6c. ADDRESS (City, State and ZIP Code): Tucson, AZ 85721	7b. ADDRESS (City, State and ZIP Code): Building 410 Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-84-0072
8c. ADDRESS (City, State and ZIP Code): Building 410 Bolling AFB DC 20332-6448	10. SOURCE OF FUNDING NOS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A2	

11. TITLE (Include Security Classification): Saguaro: A Distributed Operating System Based on Pools of Servers

12. PERSONAL AUTHOR(S): Gregory Andrews, Richard Schlichting

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 1/1/84 TO 3/12/87	88/03/25	5

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Distributed Operating Systems, Progamming Languages

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The progress achieved over the four years on the Saguaro distributed operating system is presented. The amjor accomplishments include design of the full system, prototype implementations of major system components on top of UNIX, and the implementation of portions of the system using the distributed programming language SR.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☒ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Abraham Waksman	202/767-5026	NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE

Final Technical Report

Saguaro: A Distributed Operating System Based on Pools of Servers

Grant Number: AFOSR-84-0072

Grant Duration: January 1, 1984 - December 31, 1987

Awarded to: The University of Arizona
Tucson, Arizona 85721

Principal Investigators: Gregory R. Andrews
Richard D. Schlichting
Department of Computer Science

AFOSR Program Manager: Dr. Abraham Waksman
Directorate on Mathematical and Information Sciences

G R Andrews
Gregory R. Andrews
March 25, 1988

R D Schlichting
Richard D. Schlichting
March 25, 1988

Summary

The progress achieved over the four years of the Saguaro distributed operating system project is presented. The major accomplishments include design of the full system, prototype implementations of major system components on top of UNIX, and the implementation of portions of the system using the distributed programming language SR. Substantial work was also performed on related research, including SR, the MLP system for constructing distributed mixed-language programs, the Psync interprocess communication mechanism, the *x* configurable operating system kernel, and the development of language mechanisms for performing failure handling in distributed programming languages.

Research Accomplishments

Over the past four years, we have made significant progress on the design and implementation of the Saguaro distributed operating system. Substantial work has also been performed on related research, including the SR distributed programming language, the MLP system for constructing distributed mixed-language programs, the Psync interprocess communication mechanism, the *x* configurable operating system kernel, and the design of language mechanisms for handling failures in distributed programs.

Saguaro is a new operating system for computers connected by a local-area network. Systems constructed on such an architecture have the potential advantages of concurrency and robustness. In Saguaro, these advantages are made available to the user through several mechanisms. One is *channels*, a facility that allows the input and output of different user commands to be connected to form general graphs of communicating processes. Two additional mechanisms are provided to support semi-transparent file replication and access: *reproduction sets* and *metafiles*. The advantages of concurrency and robustness are also realized at the system level by the use of pools of server processes and decentralized allocation protocols. Finally, Saguaro makes extensive use of the *Universal Type System* (UTS), a data description language to describe user data such as files and to specify the types of arguments to commands and procedures. This enables the system to assist in type checking and leads to a user interface in which command-specific templates are available to facilitate command invocation.

A number of papers have been written describing various aspects of Saguaro. An early description of the system is reported in [1], while a subsequent paper describing the design of the system appears in [2]. Another paper describing the mechanisms for enhancing file availability appears in [3]. A related paper describes the implementation and use of reproduction sets and metafiles in a UNIX environment [4]. The entire file system has now been implemented on Sun workstations; this work is described in detail in a Ph.D. dissertation [5]. Summaries of Saguaro and the related projects also appear in the proceedings of two workshops on experimental distributed systems [6]. Finally, a formal derivation of a rectangle partitioning algorithm motivated by the window manipulation necessary for the Saguaro user interface is outlined in [7].

As mentioned, significant progress has also been made on related projects. One of these is further development of the SR distributed programming language, which has been used to implement the Saguaro file system and will be used to implement the rest of Saguaro. Over the past several years, a large amount of effort was expended revising SR and implementing a compiler for the new version. The main language constructs are still resources and operations: resources encapsulate processes and variables they share, operations provide the primary mechanism for process interaction. One way in which SR has changed is that both resources and processes are now created dynamically. Another change is that the mechanisms for operation invocation—**call** and **send**—and operation implementation—**proc** and **in**—have been extended and integrated. Consequently, all of local and remote procedure call, rendezvous, dynamic process creation, asynchronous message passing, multicast, and semaphores are supported. We have found this flexibility to be very useful for distributed programming. The language has also been refined in numerous additional ways to provide additional flexibility. Moreover, by basing

SR on a small number of well-integrated concepts, the language is also relatively simple and has a reasonably efficient implementation.

The compiler and runtime support currently execute on top of UNIX; they have been in use since November 1985. Since that time, numerous small enhancements to the language have been made, mainly in response to comments from users. SR now runs on both Vaxes and Suns and most of the language features have been implemented. The implementation is currently in beta test and will soon be ready for release to whomever is interested.

The current version of the SR language is defined in [8]. An overview of the language and its implementation is given in [9], while an earlier version of the language is documented in two technical reports [10, 11]. A detailed discussion of how SR has evolved—what has changed and why as well as what has not changed and why not—is given in [12]. Much of this work was performed as part of the dissertation research of a recently completed doctoral student [13]. Experience using the language to implement the Saguaro file system appears in [5]. In addition, the variety of communication primitives provided by SR has facilitated the research of Stella Atkins, who was a visiting professor at Arizona during Spring 1986 [14, 15].

A second Saguaro-related project is the MLP system for constructing distributed, mixed language programs. This system allows users to write sequential programs in which each procedure can be written in any one of several programming languages and located on any machine in a network. In effect, MLP allows heterogeneity in language and machine functionality to be exploited by the addition of a general remote procedure call facility to each supported language. Writing an MLP program involves three steps: the addition of interface specifications written in UTS to each procedure, translation of the augmented source using an MLP translator for the given host language, and the invocation of the MLP linker to produce an executable program. At run-time, values are transmitted between procedures using a machine and language independent data representation; conversions between such values and their equivalent host language values are performed automatically in most cases. The system executes on a collection of Vaxes and Suns running Berkeley UNIX. Currently supported languages are C, Pascal, and Icon. Work has recently begun on a second version of the system.

The UTS system and its application to mixed language programming are described in [16]. A description of the implementation of MLP and our experience in using the system appears in [17]. A user's manual on the MLP system has also been written [18], as has a report describing how to add a new language to the system [19]. The way in which the object-oriented distributed programming language Emerald is being integrated into the second version of the system appears in [20].

Another line of research has involved two separate projects investigating aspects of interprocess communication. The first is Psync, an interprocess communication mechanism that supports a new abstraction through which a group of processes can exchange messages. The novel aspect of Psync is that it preserves the *happened before* partial ordering of messages exchanged among multiple processes. Just as physical clock signals are encoded with data bits in a raw communication channel to help keep the source and destination synchronized, Psync explicitly embeds timing information drawn from the distributed computation's *logical clock*.

We have recently implemented and experimented with a prototype version of Psync. The prototype demonstrates that recording the happened before relation in the communications subsystem can be implemented on an unreliable communications network at little cost. Also, the abstraction provided by Psync is general enough to support efficient and elegant implementations of a wide spectrum of communication paradigms.

The second project is the *x*-kernel, a configurable operating system kernel designed to support experimentation in interprocess communication and distributed programming. The *x*-kernel's underlying architecture provides a rich set of abstractions that are used to construct and compose communication protocols. The architecture is interesting because the these abstractions are both general enough to accommodate a wide range of protocols and efficient enough to provide a useful testbed in which protocol performance can be accurately measured.

An initial description of Psync appears in [21] (that paper was selected as the outstanding paper at its symposium). A more comprehensive paper that includes discussion of the implementation is currently nearing completion [22]. The design of the *x*-kernel is documented in [23].

The final related line of investigation has been developing language mechanisms for handling failures. One property that makes failures difficult to handle in distributed systems is that they can occur concurrently with other system events. We have been investigating an approach for writing fault-tolerant distributed programs that can cope with such asynchrony in a systematic manner. The basic idea is to treat failures as just another class of events that are handled similarly to normal system events. Linguistic constructs that can be added to distributed programming languages with minimal impact are then proposed to handle such failure events. Although our approach is applicable to most concurrent languages, to make our ideas precise we have been using the SR distributed programming language as a basis for incorporating these constructs. Two non-trivial fault-tolerant protocols have been written in this extended SR language: replicated directory management and two-phase commit. The class of failures considered are those suffered by fail-stop processors.

Our initial proposal for failure handling mechanisms is described in [24]. The mechanisms have recently been refined and extended, as described in [25].

Finally, a paper has been written that surveys several of these projects in distributed languages and systems, and offers observations based on the experience gained during their design, implementation, and use [26]. The relevant projects are the SR distributed programming language, the Saguaro distributed operating system, the MLP system for constructing distributed mixed-language programs, the object-based distributed programming language Emerald, and the Psync interprocess communication mechanism. The observations address the experimentation process itself as well as the design of distributed software.

References

[1] Andrews, G.R., Schlichting, R.D., Buchholz, N., Hayes, R., and Purdin, T. The Saguaro distributed operating system. Technical Report TR 85-9, Department of Computer Science, University of Arizona, April 1985.

[2] Andrews, G.R., Schlichting, R.D., Hayes, R., and Purdin, T. The design of the Saguaro distributed operating system. *IEEE Trans. on Soft. Eng. SE-13*,1 (Jan. 1987), 104-118.

[3] Schlichting, R.D., Andrews, G.R., and Purdin, T. Mechanisms to enhance file availability in distributed systems, *Proc. 16th Fault-Tolerant Computing Symposium*, Vienna, July 1986, 44-49.

[4] Purdin, T., Schlichting, R.D., and Andrews, G.R. A file replication mechanism for Berkeley UNIX. *Software—Practice and Experience 17*,12 (Dec. 1987), 923-940.

[5] Purdin, T. *Enhancing File Availability in Distributed Systems (The Saguaro File System).* Ph.D. Dissertation, Department of Computer Science, University of Arizona, August 1987.

[6] Andrews, G.R., and Schlichting, R.D. The Saguaro distributed operating system and related projects. *Proceeding of the SIGOPS Workshop on Making Distributed Systems Work*, Amsterdam (Sept. 1986). Also, *Proceedings of the IEEE Computer Society's Workshop on Design Principles in Experimental Distributed Systems*, West Lafayette, IN (Oct. 1986).

[7] Schlichting, R.D. Deriving an algorithm for partitioning rectangles. Technical Report TR 85-10, Department of Computer Science, University of Arizona, May 1985.

[8] Andrews. G.R., and Olsson, R.A. Revised report on the SR programming language. Technical Report TR 87-27, Department of Computer Science, University of Arizona, November 1987

[9] Andrews, G.R., Olsson, R.A., et al. An overview of the SR language and implementation. *ACM Trans. on Prog. Lang. and Systems 10*,1 (Jan. 1988), 51-86.

[10] Olsson, R.A. and Andrews, G.R. SuccessoR: Refinements to SR. Technical Report TR 84-3, Department of Computer Science, University of Arizona, March 1984

[11] Olsson, R.A. and Andrews, G.R. An implementation of SuccessoR. Technical Report TR 84-4, Department of Computer Science, University of Arizona, March 1984

[12] Andrews, G.R. and Olsson, R.A. The evolution of the SR language. *Distributed Computing 1*,3 (July 1986), 133-149.

[13] Olsson, R.A. Issues in distributed programming: The evolution of SR. Ph.D. Dissertation, Department of Computer Science, University of Arizona, Aug. 1986.

[14] Atkins, M.S and Olsson, R.A. Performance of multi-tasking and synchronization mechanisms in the programming language SR. *Software—Practice and Experience* (1988), to appear.

[15] Atkins, M.S. Experiments in SR with different upcall program structures. *ACM Trans. on Computer Systems*, to appear.

[16] Hayes, R., and Schlichting, R.D. Facilitating mixed language programming in distributed systems. *IEEE Trans. on Soft. Eng. SE-13*,12 (December 1987), 1254-1264.

[17] Hayes, R., Manweiler, S.W., and Schlichting, R.D. A simple system for constructing distributed, mixed language programs. *Software—Practice and Experience* (1988), to appear.

[18] Manweiler, S.W., Hayes, R., and Schlichting, R.D. The MLP system user's manual. Technical Report TR 86-4, Department of Computer Science, University of Arizona, February 1986.

[19] Manweiler, S.W., Hayes, R., and Schlichting, R.D. Adding new languages to the MLP system. Technical Report TR 86-16, Department of Computer Science, University of Arizona, May 1986.

[20] Hayes, R., Hutchinson, N.C., and Schlichting, R.D. Integrating an object-oriented programming language into a system for mixed-language programming. Submitted for publication, March 1988.

[21] Peterson, L. L. Preserving context information in an IPC abstraction. *Proc. 6th Symposium on Reliability in Distributed Software and Database Systems* (March 1987), 22-31 (Outstanding Paper Award).

[22] Peterson, L.L., Buchholz, N., and Schlichting, R.D. Preserving and using context information in interprocess communication. In preparation.

[23] Hutchinson, N.C., and Peterson, L.L. Design of the *x*-kernel. Submitted for publication, March 1988.

[24] Schlichting, R.D. and Purdin, T. Failure handling in distributed programming languages. *Proc. 5th Symposium on Reliability in Distributed Software and Database Systems*, Los Angeles, Jan. 1986, 59-66.

[25] Schlichting, R.D., Cristian, F., and Purdin, T. Mechanisms for failure handling in distributed programming languages. Submitted for publication, May 1987.

[26] Schlichting, R.D., Andrews, G.R., Hutchinson, N.C., Olsson, R.A., and Peterson, L.L. Observations on building distributed languages and systems. *Experiences with Distributed Systems*, (J. Nehmer, Ed.), *Lecture Notes in Computer Science*, Springer-Verlag, New York, to appear.

Participating Professionals

Gregory R. Andrews, Professor and Department Head, Dept. of Computer Science, University of Arizona

Richard D. Schlichting, Assistant Professor, Dept. of Computer Science, University of Arizona

Larry L. Peterson, Assistant Professor, Dept. of Computer Science, University of Arizona

Ronald A. Olsson, Assistant Professor, Division of Computer Science, University of California, Davis (Ph.D. earned at Arizona, 1986)

Titus Purdin, Assistant Professor, Dept. of Computer Science, Colorado State University (Ph.D. earned at Arizona, 1987)

Stella Atkins, Assistant Professor, Dept. of Computer Science, Simon Fraser University (visiting Assistant Professor, 1986)

Nick Buchholz, Ph.D. student, Dept. of Computer Science, University of Arizona

Shyamal Chowdhury, Ph.D. student, Dept. of Computer Science, University of Arizona

Mike Coffin, Ph.D. student, Dept. of Computer Science, University of Arizona

Irv Elshoff, Ph.D. student, Dept. of Computer Science, University of Arizona

Roger Hayes, Ph.D. student, Dept. of Computer Science, University of Arizona

Janalee O'Bagy, Ph.D. student, Dept. of Computer Science, University of Arizona

Kelvin Nilsen, Ph.D. student, Dept. of Computer Science, University of Arizona

Ajei Gopal, Ph.D. student, Dept. of Computer Science, Cornell University (M.S. earned at Arizona, 1985)

Steve W. Manweiler, Member, Technical Staff, Hewlett-Packard Corp., Ft. Collins, Colorado (M.S. earned at Arizona, 1986)

END

DATE

FILMED

6-1988

DTIC